CHAPTER 2

INTRODUCTION TO BASIC

Fundamental Concepts and Language Structure

BASIC is different from other programming languages in its concern for the inexperienced user. Although it is a general-purpose programming language, it is designed primarily to be easy to learn, easy to use, and easy to remember. BASIC is oriented toward, but not restricted to, interactive use. Its statement structure is kept simple, and special rules are kept to a minimum. A BASIC program is meant to be simple so that even a novice is able to determine what the program is expected to do on the basis of examination. Only a little knowledge of BASIC is required to solve simple problems.

Simple BASIC programs can be used much like a small calculator. You might want to ask the computer to multiply 4 times 25 and print the results. The following example shows you how to accomplish this:

```
10  PRINT 4*25
20  END
RUN
 100
```

You keyed three lines into the computer. In the first line, line number 10, you told the computer to calculate 4 times 25 and print the answer (* in BASIC means multiply). In the second line, line number 20, you told the computer there were no more instructions in the program. The last command you gave the computer was **RUN,** a system command which tells the computer to execute the instructions and give the results. The computer multiplied 4 times 25 and gave the answer, which is 100.

As simple as it may seem, this is a computer program. The same function which took only one statement in BASIC, would take several lines of coding in other high-level programming languages such as COBOL or FORTRAN.

STATEMENT STRUCTURE

The instructions which are preceded by line numbers are called statements. A complete set of statements to solve a problem is called a program. The very last statement in each program must be the **END** statement.

Program statements in the BASIC language can be constructed in free form. The various parts must all appear, however, and be given in a definite order, as shown below:

10	LET	A = 5
statement number	keyword	descriptive information to support the keyword

STATEMENT NUMBER.—(Frequently referred to as line number). This number has two vital functions: (1) to identify the statement itself (statement label); and (2) to indicate to the BASIC compiler (interpreter) where you want this statement placed in the program sequence. The statement number must be an integer (a whole number—no decimal parts or fractions). For the purpose of this text, the range for statement numbers is from 1 to 99999. However, the number of digits may vary depending on the computer you are using. Table 2-1 shows examples of statements with valid and invalid statement numbers.

Table 2-1.—BASIC Statement Numbers

Statements with valid statement numbers	Statements with invalid statement numbers	Explanation
10 LET A = 5	−10 LET A= 5	Cannot contain minus sign
40 INPUT I,Y,L	+99 INPUT I,Y,L	Cannot contain plus sign
99 PRINT I,Y,L	5.99 PRINT I,Y,L	Cannot contain decimal point
99999 END	999999 END	Too large

BASIC LANGUAGE KEYWORD.—Keywords are used to tell the computer what function is to be performed by this statement. For example, LET, PRINT, INPUT, and END as shown in Table 2-1. These and other keywords will be introduced and discussed as appropriate throughout the remaining chapters. The spelling of each keyword must be exact, or the compiler (interpreter) will tell you this is an invalid statement.

DESCRIPTIVE INFORMATION.—This information completes the description of the function to be performed and varies with the keyword used. See the first three examples in Table 2-1. There area few instances where no additional information is required or allowed. See the last example in Table 2-1.

In order to write these statements, you must first know the syntax; that is, the characters and symbols used to construct the statements, as well as the rules, conventions, and special features of the language. This includes the character set and the methods used to represent numbers and predefined functions.

The BASIC Character Set

There are three types of characters used in BASIC. These are: (1) alphabetic, (2) numeric, and (3) special characters.

ALPHABETIC CHARACTERS.—The alphabetic characters used in BASIC are the standard English alphabet, A through Z.

NUMERIC CHARACTERS .—The numeric characters used in BASIC are the digits 0 through 9.

SPECIAL CHARACTERS.—The following are special characters used in BASIC:

	Blank	‘	Single quotation mark
=	Equal sign or assignment symbol	‘‘	Double quotation mark
+	Plus sign	;	Semicolon
–	Minus sign	:	Colon
*	Asterisk or multiply symbol	!	Exclamation symbol
/	Slash or divide symbol	?	Question mark
↑	Up-arrow or exponentiation symbol	&	Ampersand
)	Right parenthesis	<	Less than symbol
(	Left parenthesis	>	Greater than symbol
,	Comma	#	Number or pound sign
.	Point or period	$	Dollar sign
		%	Percent sign

OTHER SPECIAL CHARACTERS.—Some special characters are combined to form other elements in BASIC. The following list shows these combinations:

> = greater than or equal

< = less than or equal

<> not equal

** exponentiation

BASIC Numbers

When you are using numeric data in a BASIC program, there are certain conventions that must be adhered to. You cannot use “$” (dollar sign), ‘‘, ”

(comma), or the "/" (slash) in a BASIC number. There are also restrictions on the number of digits that can be used in one data element. The number of digits may vary depending upon the computer you are using. Refer to your user's manual for specific instructions. Table 2-2 shows examples of correctly and incorrectly coded BASIC numbers.

Table 2-2.—BASIC Numbers

Valid BASIC Numbers	Invalid BASIC Numbers	Explanation
99.95	**$99.95**	Dollar sign cannot be used
100000.00	100,000.00	Comma cannot be used
+5678901.2	+567890100	Too many digits, only eight are allowed (decimal points, positive and negative signs do not count).
.25	1/4	Slash cannot be used; however, this is a valid expression as an arithmetic operation (1 divided by 4).

SCIENTIFIC NOTATION.—As seen in Table 2-2, in the third example, we have a number, +567890100, with too many digits. You may ask, how do we represent very large and very small numbers? Scientific notation is used. In scientific notation numbers are expressed in terms of a figure between 1 and 10 times a power of 10. The number 567890100 would be written 5.678901×10^8 in scientific notation. This method uses exponent form and is commonly called E notation, E form or E format. In BASIC, E notation is formed by adding the letter E and a positive or negative integer to a BASIC number. For example, E8 means "times 10 to the 8th power."

$$5.678901E8 = 5.678901 \times 10^8 = 567890100$$

Table 2-3 shows other examples of the use of E notation.

Table 2-3.—E Notation

Number using E notation	Interpreted as	Explanation
5.5E8	550,000,000	The decimal has been moved eight places to the right
5.5E-8	0.000000055	The decimal has been moved eight places to the left
–2.54321E10	–25,432,100,000	The decimal has been moved ten places to the right
–2.54321E-5	–0.0000254321	The decimal has been moved five places to the left

Since we're multiplying by powers of 10, the integer following the E indicates how many positions and in what direction to move the decimal point. If the integer after the letter "E" is negative, you move the decimal point to the left; if the integer is positive, you move the decimal point to the right.

Predefine Functions

Some mathematical functions, for example, square root and tangent, are used so frequently that they have been incorporated into the BASIC language as predefine functions. You can use these to compute the same mathematical function with many different values. Rather than writing the coding required to do each calculation, you may use the "function" capability of the BASIC language. For example:

```
LET X = SQR (36)
```

This will cause the computer to calculate the square root of 36 and assign that value to X. Table 2-4 shows a list of some of these predefine mathematical functions.

Table 2-4.—Predefined Functions

FUNCTION	DESCRIPTION
ABS (X)	Absolute value of X
ATN (X)	Arc tangent of X in radian measure
COS (X)	Cosine of X in radian measure
EXP (X)	Natural exponential of X
INT (X)	The largest integer not greater than X
LOG (X)	Natural logarithm of X (base e)
RND (X)	Generates random numbers between 0 and 1
SGN (X)	Algebraic "sign" of X: -1 if X<0, 0 if X=0, and +1 if X>0
SIN (X)	Sine of X in radian measure
SQR (X)	Square root of X
TAN (X)	Tangent of X in radian measure

FUNDAMENTAL CONCEPTS

Up to this point, we have been discussing the mechanics of the BASIC programming language. We will now discuss fundamental concepts and how

you use these in your interaction with the computer in keying in and running your programs. This will include more about statement numbers, spacing within statements, keying statements into the computer, methods of correcting mistakes, how the computer responds when your program contains a syntax error, and the use of the **REMARK** statement.

Assigning Statement Numbers

Duplicate statement numbers are not allowed. If two statements are entered with the same statement number, the computer will accept the second statement with the duplicate statement number and replace the first.

When assigning statement numbers, it is a good idea to increment them by 10; this will allow you to insert additional statements between existing statements in your program later. This is not mandatory, but it is a good practice. This technique will prevent you from having to completely renumber a program, if you find you need to add a statement after you have completed writing your program. The following example shows how this technique works:

```
10   LET A = 2
20   LET B = 4
30   LET C = 6
40   LET X = (A+C)/B
50   LET Y = (A*C)/B
60   LET Z = A+B+C
70   END
65   PRINT X,Y,Z
```

In this example the PRINT statement, which is needed to print the results from your calculations was omitted. Since the line numbers were incremented by 10, it was easy to assign the PRINT statement a line number between 60 and 70.

When the program is run or you list your program, the PRINT statement, line number 65, will be inserted in its proper place by the computer.

```
10   LET A = 2
20   LET B = 4
30   LET C = 6
40   LET X = (A+C)/B
50   LET Y = (A*C)/B
60   LET Z = A+B+C
65   PRINT X,Y,Z
70   END
RUN
 2             3             12
```

Spacing Within Statements

Spacing within statements in your program is not very important, the computer generally ignores spaces except those within quotation marks. For example, one of the sample programs used earlier could be written this way:

```
10PRINT4*25
20END
```

OR:

```
10   PRINT    4    *   25
20   END
```

or various other ways. However, as a matter of practice, you will want to use spacing that provides clarity and maintains the integrity of the statement. Appropriate use of spaces within a line will make the line easier for you and others to read and understand.

The sample program below is an example of the recommended spacing within statements:

```
10   REMARK THIS PROGRAM CONVERTS INCHES TO CENTIMETERS
20   INPUT 1
30   LET C = 1*2.54
40   PRINT "INCHES","CENTIMETERS"
50   PRINT I,C
60   END
```

In this example, we have used spacing techniques similar to those you would use in ordinary typing.

KEYING IN A PROGRAM

Once you have written a program and have a computer available, you are ready to key in your BASIC program. The procedures for this will vary, depending upon the computer system you are using. Consult your user's manual for procedures for your specific computer.

Keying Statements

To enter BASIC statements, you key in the statement beginning with the statement (line) number, after the prompt on the terminal. Once you have completed the statement, depress the **RETURN** key. Only one statement may be entered per line. Some computers are equipped with an automatic line numbering feature which will automatically number your statements, incrementing them by 10.

It is not necessary to key in your BASIC statements in sequence by line number. The computer will sort the statements by line number and place them in ascending sequence, regardless of the sequence in which you keyed them in. The following example shows what we mean:

```
10    REMARK THIS PROGRAM CONVERTS INCHES TO CENTIMETERS
20    INPUT I
30    LET C = I*2.54
40    PRINT ''INCHES'',''CENTIMETERS''
60    END
50    PRINT I,C
LIST
```

You forgot to type in line 50 when you initially entered your program, but that is not a catastrophe. You can enter it after line 60. By entering the system command **LIST,** you will find line 50 has been placed in the proper sequence in the program. Your listing will look like this:

```
10    REMARK THIS PROGRAM CONVERTS INCHES TO CENTIMETERS
20    INPUT I
30    LET C = I*2.54
40    PRINT ''INCHES'',''CENTIMETERS''
50    PRINT I,C
60    END
```

Correcting Mistakes

At one time or another, we all make mistakes when we're keying in statements. For that reason, ways are provided for you to make corrections. These may vary depending on the computer you're using. On many computers, the statements you key in are not sent to the interpreter until you have depressed the RETURN key. Should you make a mistake in a statement before you have depressed the return key, you can backspace with the backspace key and key the correct information.

For example:

you typed

```
10    LET A = B+
```

but you intended to type an asterisk (multiply) instead of a plus sign (addition). You can correct it by backspacing once and keying the correct information. The * (asterisk) will replace the + (plus), then you continue by keying the rest of the statement.

Should you make more than one mistake in a line, use one backspace for each character you want to erase or replace.

If you make a mistake in a statement and you have already entered the statement, you will have to correct it in another way. The following examples show how it can be done.

Suppose you had entered the following program:

```
10  LTE A=2
20  LET B=4
30  LET C=6
40  LET X=(A+C)/B
50  LET Y=(A*C)/B
60  LET Z=A+B+C
70  PRINR X,Y,Z
80  END
```

Lines 10 and 70 contain errors. You can correct them by simply retyping the lines as they should be.

```
10  LET A=2
70  PRINT X,Y,Z
```

These will overlay (replace) the original lines 10 and 70. If you list or display your program after making these changes, it will appear as follows:

```
10  LET A=2
20  LET B=4
30  LET C=6
40  LET X=(A+C)/B
50  LET Y=(A*C)/B
60  LET Z=A+B+C
70  PRINT X,Y,Z
80  END
```

Lines 10 and 70 which were originally entered incorrectly were replaced by the retyped lines and are in correct form and in proper sequence in the

program. As we stated earlier, duplicate line numbers are not allowed and the second statement entered with the same line number will overlay or replace the first.

Some computers have an EDIT/RECALL feature which allows you to enter the line number you want to correct and then depress **EDIT/RECALL.** It will display this line on the screen and allow you to make changes to it.

You may also delete a line by typing the command **DELETE** and the line number, for example:

```
DELETE  50
```

This will delete line 50.

Another way to delete a line is to type in only the line number and depress the RETURN key, for example:

```
50
```

This will also delete line 50.

When you are keying in your program, some computers will check for syntax errors as each statement is entered. Others will check for syntax errors when you enter the system command RUN. In either case, you'11 get an error message indicating where there is a syntax error. (See the following example.) These errors must be corrected before the program can be executed.

```
10  LET W=2
20  LET X=4
30  LET Y=6
40  LET Z=8
50  PRINT W+6,X*Y,Z-X,W/X
60  PRINT W,X,)Y,Z
70  END
RUN
LINE 60 SYNTAX ERROR IN EXPRESSION
```

Look at line 60, find the error. The **)Y** is incorrect; the **)** should not be there. Now that you know what the error is, you can use one of the methods discussed to correct it and try again.

Unfortunately, at this stage, the computer can only find syntax errors, it cannot determine if you have made a logic error. Logic errors will be found when you try to run the program during debugging and testing.

To help find logic errors, you may find it useful to include some documentation about the program within it. To do this, you use the REMARK statement.

REMARK Statement

All languages provide the capability for inserting programmer comments to make the program listing more readable and to aid in testing and documenting the program. In BASIC, the keyword **REMARK** (which may be abbreviated **REM)** is used. The REMARK statement is a nonexecutable statement; that is, the REM and following comments appear only in the listing of the program and do not, in any way, affect execution of the program.

FORMAT:

```
10      REMARK      THIS PROGRAM CONVERTS INCHES TO CENTIMETERS
```

statement number — keyword — any programmer comments

When a statement with the keyword REMARK is used, the entire line on which the statement appears is considered a comment.

The following example shows how the REMARK statement is used to document a program so that anyone can understand what the program is used for:

```
10  REMARK THIS PROGRAM IS USED TO CONVERT INCHES
20  REMARK TO CENTIMETERS. WHEN THE SYSTEM
30  REMARK ASKS FOR INPUT, INPUT THE NUMBER OF
40  REMARK INCHES YOU WANT CONVERTED TO CENTIMETERS
50  INPUT I
60  LET C = I*2.54
70  PRINT ''INCHES'',''CENTIMETERS''
80  PRINT I,C
90  END
```

Not only can REMARK statements be used at the beginning of a program, they can be used throughout the program to separate different segments of a program and to introduce subroutines or loops.

SUMMARY

BASIC is designed to be easy to learn, easy to use, and easy to remember. BASIC programs are meant to be simple so that even a novice can understand them.

Instructions which are preceded by line numbers are called program statements. The parts of a statement are: *statement (line) number, keyword, and descriptive information.*

The BASIC character set is divided into three categories: alphabetic, numeric, and special characters. E notation (scientific notation) is used for representing very large and very small numbers. BASIC has included several predefined functions so that you don't have to code some of the more common mathematical functions.

Line numbers tell the computer the sequence of the instructions in your program. It is a good practice to increment your line numbers by 10 to allow for inserting additional statements between existing statements in your program. Spacing within a line is not important; however, appropriate use of spaces within a line makes it easier for you and others to read and understand. The first step in entering in a statement is keying in the line number.

You do not have to key the BASIC statements in sequence, the computer sorts them into line number sequence. The system command **LIST** causes your program to be listed with the statements in ascending sequence by line number. The system command **RUN** is used to tell the computer to execute the program.

Unfortunately the computer does not detect logic errors; you will have to find those through testing and debugging.

The **REMARK** statement is used to aid in documenting your program.

CHAPTER 2

EXERCISES

1. Arrange the following statement elements in proper order according to the BASIC statement structure and label the parts.

```
A,B,C   40   PRINT
```

2. Which of the following are correctly coded BASIC line numbers? For those incorrectly coded, what is wrong with them?

A. 20	F. 54321
B. 25	G. 123456
C. 30	H. 99999
D. 1/4	I. #9999
E. 9.99	J. .2500

3. Convert the following numbers in E notation into the numbers they represent.

A. 2.5E3
B. –1.5E2
C. 9.9E-5
D. –3.5E6

4. Which of the following are correctly coded numbers? For those incorrectly coded, what is wrong with them?

A. 95	E. .95
B. 9,500	F. 52
C. $100.00	G. 65$
D. 1/4	H. 74

5. A. If you typed the following BASIC program into the computer and gave the system command LIST, what would the listing look like?

```
10   LET A=2
20   LET C=6
30   PRINT A+B+C
20   LET C=8
15   LET B=4
40   END
25   PRINT "TOTAL"
```

B. If you then typed the system command RUN, what would be printed?

CHAPTER 2

EXERCISE ANSWERS

1. 40 PRINT A,B,C,

   line #, keyword, descriptive information

2. correctly coded line numbers

   A,B,C,F,H

   incorrectly coded line numbers

   D. 1/4 must be a whole number
   E. 9.99 cannot contain a decimal point
   G. 123456 maximum number allowed is 99999
   1. #9999 cannot contain special characters
   J. .2500 cannot contain a decimal point

3. A. 2500.0
   B. -150.0
   C. 0.000099
   D. -3500000.0

4. correctly coded numbers

   A,E,F,H

   incorrectly coded numbers

   B. 9,500 cannot contain commas
   C. $100.00 cannot contain $
   D. 1/4 cannot contain special characters except decimal points
   G. 65$ cannot contain $

5. A. 10 LET A = 2

   15 LET B = 4

   20 LET C = 8

   25 PRINT "TOTAL"

   30 PRINT A+B+C

   40 END

   B. TOTAL

   14